

(19)



Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number:

0 383 474
A2

(12)

EUROPEAN PATENT APPLICATION

(21) Application number: 90301251.6

(51) Int. Cl.⁵: G06F 13/26

(22) Date of filing: 06.02.90

(30) Priority: 13.02.89 US 310409

(43) Date of publication of application:
22.08.90 Bulletin 90/34(84) Designated Contracting States:
DE FR GB

(71) Applicant: International Business Machines Corporation
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: Chang, Tai-Lin
19861 Greystone Lane
San Jose, CA 95120(US)
Inventor: Hunter, Paul Wayne
1270 Chateau Drive
San Jose, CA 95120(US)
Inventor: Lang, Donald John
11220 Monterey Court
Cupertino, CA 95014(US)
Inventor: Luning, Stephen Gouze
5966 Thorntree Drive
San Jose, CA 95120(US)

(74) Representative: Grant, Iain Murray
IBM United Kingdom Limited Intellectual
Property Department Hursley Park
Winchester Hampshire SO21 2JN(GB)

(54) Processor interrupt control.

(57) In a programmed machine, such as an peripheral controller, programmed operations are executed in a one of several operational contexts. Each context may be initiated by a corresponding interruption signal. Any context which has been activated remains active until quiesced by program execution. One of the active contexts is a current context in which all instruction executions are currently occurring. In each cycle of the programmed machine, all active contexts and received and stored interruption signals, each for respective ones of the contexts, are compared to find the context highest priority context. Such highest priority context is compared with the current context priority for determining whether or not the programmed machine should change current contexts.

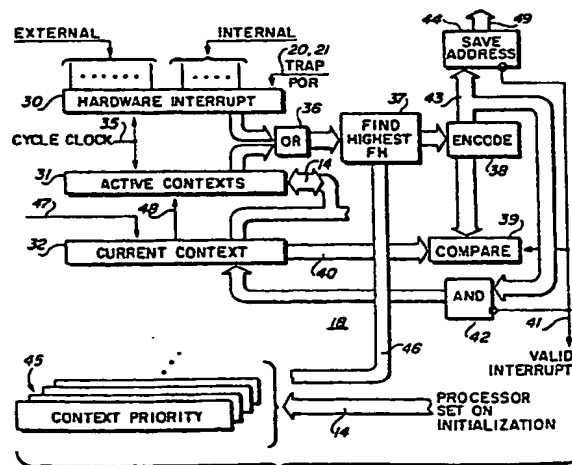


FIG. 2

PROCESSOR INTERRUPT CONTROL

The present invention relates to processor interrupt control, bearing in mind that, in this context, the term "processor" includes a peripheral controller for forming part of a greater data processing complex.

Programmed machines of all types have included both internal, external and programmed interruption facilities. Some of the facilities have had fixed priority of interruptions, such as round robin interruption priorities and the like. Other interruption facilities have provided variable priorities of interruption.

Peripheral controllers typically have used a fixed priority of interruption which is either plugged into the controller in a plugboard (also referred to as a pinboard) or is wired into the electronic circuitry. In such controllers, the position of connection of peripheral machines indicates priority and connections of internal circuits indicate their respective priorities.

Thacker in US patent 4,103,330 shows a data processing apparatus having a priority interruption encoder 28. The interruption encoder receives interruption or wake up requests from I/O controllers. This encoder encodes the interrupt requests into a 4-bit quantity which in turn is applied to a current task register 36. This quantity indicates the current highest priority outstanding request. Such priority is fixed by the electronic circuits of the encoder 28 illustrated in this document. It is desired to provide a more flexible and effective interruption evaluation and processing.

Byrns in expired US patent 3,599,162 shows an interruption handling apparatus have a plurality of interruption tables. Each table can store a plurality of received interrupt requests all of which have the same priority; each table storing interrupts of different priorities. Apparently the interrupt requests in any given table are handled on a FIFO basis. As such, the FIFO selection within a table is a tie breaking function between interrupt requests having the same priority.

Yokoyama in US patent 4,338,662 shows a microinstruction interruption control in which microinstructions which are interrupted are stored in a stack, apparently in accordance with their priority ratings. The interrupted microinstruction executions are resumed whenever the interrupting microinstruction finishes. This scheme also appears to use a fixed priority.

Hodge in US patent 4,636,944 shows a multi-level priority microinterrupt controller. Only one interrupt signal for each of the interrupt levels can be active at one time. A higher priority interrupt causes the controller to stack the currently inter-

rupted instruction for enabling the interrupted instruction to be resumed when the interruption has been completed.

Branigin et al in US patent 4,358,829 show a dynamic rank ordered scheduling mechanism. The mechanism allows insertion of elements into a list of positions determined by the rank of the elements and allows deletion of elements only from the top of the list. The arrangement provides for dynamic aging of the priorities of the elements in the list.

The present invention provides data processor interrupt logic which separately maintains stored a record of the priorities of processor contexts, which contexts are active, which context is the current executing context and which contexts are requesting to interrupt the current context and continuously compares the priorities of the current context and the highest priority requesting active context as determined by combining the stored records, validating - enabling - the interrupt if its priority is the higher.

Such an arrangement provides interruption control in which the executing program of instructions needs no awareness of the current executing context priority nor the priority of any active context which is not currently executing. It is possible to enable altering priority of the interruption by the state of the context as opposed to the identity of the context, that is, to enable interrupt control in accordance with the indicated highest priority interrupt context.

The present invention also provides a processor including such an arrangement and the counterpart method of interruption control in a processor.

As disclosed hereinafter, interruption signals are analysed for determining whether or not a programmed processor is to be interrupted. The machine establishes a plurality of operational contexts for the programmed processor together with primary and secondary priorities of interruptions for each of the contexts.

A means stores one or more interruption signals which indicate requests for interruption of the processor operation for changing instruction execution to a context corresponding to the stored interruption signals, respectively.

The machine also maintains an indication of all active contexts in which the programmed processor may or may not be currently executing instructions and identification of the current context in which the programmed processor is executing instructions. During predetermined cycles of operation of the programmed processor, means compare all of

the active contexts and stored interruption signals with each other including ascertaining the primary and secondary priorities of each for indicating a highest priority active context or interruption signal.

The highest priority of the indicated highest priority active context or interruption signal with the current context in both primary and secondary priorities and interrupting the programmed processor when the indicated highest priority active context or interruption signal is higher than the priority of the current context.

None of the above-discussed documents suggests a flexible but efficient interruption procedure or apparatus which is based upon an executing context, which enables quiescing any active context, whether executing or not, and for tie breaking between any interrupts for contexts having the same priority of interruption.

The present invention will be described further by way of example with reference to an embodiment thereof as illustrated in the accompanying drawings in which:-

Fig. 1 is a simplified block diagram of a programmed processor for use in a peripheral controller and which is incorporated into a single semiconductor chip;

Fig. 2 diagrammatically illustrates a context and interrupt control or manager for use in the Fig. 1 illustrated programmed processor;

Fig. 3 is a simplified flow diagram showing response to a valid interrupt and program initiated context quiescing in the Fig. 1 illustrated processor;

Fig. 4 is a simplified machine operations chart showing operation of the Fig. 2 illustrated apparatus;

Fig. 5 is a simplified machine operations chart showing context quiescing when using the Fig. 2 illustrated context manager; and

Fig. 6 is a simplified machine operations chart illustrating finding the highest priority active context or context related to an interruption signal by the Fig. 2 illustrated context manager.

Referring now more particularly to the appended drawing, like numerals indicate like parts and structural features in the various figures. Fig. 1 illustrates a programmed processor system which is a part of a peripheral controller (not shown) integrated into a single semiconductor chip 10. The system includes a chip processor 11 which consists of microprocessor circuits of any usual design. An instruction store 12 is located off-chip and is accessible by processor 11 for fetching instructions to be executed. A set of storage registers 13 (which can be a RAM) in chip 10 are connected to chip processor 11 over in-chip bus 14. Bus 14 has the usual plurality of electrical conductors, as is known, some of the conductors carry data signals while other conductors carry tag or control signals.

Such an arrangement can also be considered as two buses, a data bus and a control bus. Registers 13 act as a local store for processor 11. Included in registers 13 is a program save area 15 used as later described for program storage word (PSW) swaps. A plurality of bus adapters 16 operatively connect in-chip bus 14 to a plurality of other functional chips (such chips, not shown, operatively connect the respective bus adapters 16 to other functioning chips which in turn are connected to host processor channels, peripheral devices and the like) on a circuit board (not shown), to other circuit boards (not shown) or other devices (not shown) as may be appropriate. Interrupt circuits 18 are connected to in-chip bus 14 for communication with chip processor 11 for interrupting the operation of same, as is known in the art.

Interrupt circuits 18 receive diverse program execution interrupting signals. A highest priority of interrupt, which is fixed in the illustrated embodiment, is a power on reset (POR) received over external chip connection 20 from an external source (not shown) indicating that a power reset is occurring. Another external interrupt of fixed highest priority is a trap signal received over external chip connection 21. The response of chip processor 11 to these two types of interruptions is known. A set of external interrupt input chip connections 22 connect other chips, such as referred to above and which are connected to one or more of the data buses 17 over which the processor system on chip 10 communicates. A set of chip internal interrupt inputs 23 also go into circuits 18. The aspect of the above which is material to the present invention relates to the handling of all of the above-identified interruption signals and managing the context in which chip processor 11 executes streams of instructions.

Fig. 2 is a simplified showing of interrupt circuits 18. Hardware interrupt register 30 is a double-rank register which asynchronously receives interruption request signals in a first rank (not shown). Register 30 is a phase-hold register which holds its signal contents for one interrupt evaluation cycle; about one cycle of machine operation. The interrupt signals inputted to the register 30 either can remain active until the interrupt is honoured or can be reissued each machine cycle until the interrupt is honoured. Each time a cycle of chip processor 11 reaches a predetermined elapsed time from the beginning of a cycle, the request signals in a first rank (not shown) of register 30 are transferred to its second rank (not shown). The later-described circuits effectively scan the second rank of register 30 for reading the received interrupt requests. There is one bit position in register 30 corresponding to a respective context, as will become apparent.

Chip processor 11 executes in only one context of a plurality of possible contexts. Each of the contexts may be set up and entered into by chip processor 11 under the management and control of interrupt circuits 18. The priority of interrupt is set, as later described, by program or software means. Such setting during a context execution reflects the current state of the context. That is, a designer can incorporate into the programming or software controls beyond the present description which alter the interrupt priorities in accordance with the design. The Fig. 2 illustrated apparatus and system enable this flexibility. Also, the executing program of chip processor 11 has no need to know which of the contexts it is executing in nor how many nor which contexts are active (work is still to be done in those context which are active). All of this control and management is performed by interrupt circuits 18 with minimal software interaction in chip processor 11. Active context register 31 stores identifications of all the currently active contexts. Such identifications are bit positions in the register which correspond to the respective context; one bit per context. A set bit indicates the corresponding context is active while a reset bit indicates the corresponding context is not active, i.e. quiesced. This storage includes the currently executing active context, such executing context is also identified by a corresponding bit being set in current context register 32, as will become apparent. Processor 11 sets and senses the contents of active context register 31 via bus 14 while later-described circuits set current context register 32; that register can be sensed or written to by chip processor 11. The later-described valid interrupt handling and quiescing context handling involves chip processor 11 to the extent that it updates the contents of active context register 31. This action does not imply that program execution within a given context requires any information as to which contexts are active.

At a predetermined point in each cycle of chip processor 11 operations, a cycle clock signal from processor 11 travels over line 35 to actuate the register 30 second rank and the active context register 31 to transfer a copy of their respective contents through a set of logic-OR circuits 36, each such logic-OR circuit merges like positioned bits of the two registers, i.e. bit signals logically associated with the same context, and passes the merged bit signals to find highest circuit 37, hereafter FH circuit 37. FH circuit 37 staticises the merged bit signal in a suitable register (not shown) for later-described processing. Such processing identifies which of the merged bit signals (either active context or interrupt request) have the highest found priority of interruption. Following identification of the highest priority active context or received request, FH circuit indicates to encode circuit 38 which bit (i.e. which context) has the highest priority for a priority comparison between the identification of the current context of register 32

priority. Firstly, the bit is encoded by encode circuit 38 into an address quantity identifying the save area 15 address which is to store the PSW for the identified highest found priority. Compare circuit 39 receives the bit indicating the current context over bus 40 from register 32. Compare circuit 32 encodes the bit into save area 15 address for the current context. The save area 15 addresses for the context PSW's are arranged in a predetermined order. When compare circuit 39 finds the address from encode circuit 38 is different from the address received from current context register 32 and the highest found interrupt has a higher priority than the current context priority, a valid interrupt signal is generated. The valid interrupt signal is supplied over line 41 to chip processor 11. Substantially simultaneously, the contents of current context register 32 are updated via AND circuits 42 as enabled by the valid interrupt signal to the context associated with the found highest priority. Alternately, chip processor 11 may update the contents of current context register 32 to reflect the upcoming PSW swap. Also address register 44 is enabled to receive the new save area 15 address for enabling chip processor 11 to effect a known PSW swap between the two contexts, the interrupting and interrupted contexts. At this time the active context register 31 is updated via bus 14 by chip processor 11 to reflect honouring the interrupt which activates a quiesced context. The bit position in the register 30 second rank corresponding to the interrupting context (found highest-priority context) is reset by the phase-hold time expiring.

The operation of FH circuit 37 combines the identification of the contexts received from registers 30,31 with assigned context priorities stored in the context priority registers 45. Preferably during system initialisation (such as POR), the chip processor accesses the registers 45 via bus 14 for setting same to initially and dynamically assign priorities to the contexts. The dynamic assignment reflects the current state of the current executing context. In some instances the current executing context may not dynamically change all of the context priorities. Some of the context priorities may still reflect the state of an earlier executing context which has either finished or was interrupted. Each of the registers 45 have a predetermined number of bit positions, one bit position for each of the contexts. The lower number the bit position in each register, the higher the priority that position indicates. The lowest numbered and highest numbered bit positions are not settable, such positions are respectively reserved for traps and for background processing. During a POR the contents

of registers are cleared requiring chip processor 11 to again establish initial assigned priorities. The lowest numbered bit which is set in the respective registers 45 indicate the priority for the corresponding context. If no bit is set in any one register, then that context has no priority assigned, i.e. if no external unit is attached which would cooperate with such a context or for some reason the interruption by a context is to be inhibited or masked for reasons beyond the present description, then erasing or not setting the respective one of the registers 45 effectively masks out interrupts intended for that context. In various ones of the context priority registers, some of the bit positions may not be allowed to be assigned to such context. This control masks certain priorities from a context, such selections are a matter of design choice.

FH circuit 37 for each of the active contexts and stored interrupt requests received via logic-OR circuits 36, accesses the respective one of the context priority registers 45 for ascertaining the priority assigned to the context. The scan of registers 45 reveals which of the contexts being evaluated has the highest assigned priority to enable FH circuit 37 to emit the above-described signal to encode circuit 38. It is to be understood that several designs and procedures may be employed for evaluating the priority scanning within circuit FH 37. The Fig. 6 machine operations chart illustrates the logic design of FH circuit 37.

The software interaction of chip processor 11 with the interrupt processing is shown in Fig. 3 in extremely simplified form. In responding to a valid interrupt signal on line 41, chip processor 11 in step 50 responds to the line 41 signal to read register 44 in preparation for a PSW swap in the next machine cycle at step 51. No instruction is fetched from instruction store 12 for step 51. Following step 51 in the second machine cycle of the interruption processing, chip processor 11 is operating in the interrupting context. When chip processor 11 has completed processing for a given context, that context is quiesced at step 52. In this step, chip processor 11 supplies a quiesce signal over line 47 to interrupt circuits 18 to set current context register 32 to background processing (cleared) and reset the associated active bit in active context register 31 such that during the cycle of circuit 18 operation any current highest priority active context or received interrupt request is able to become the interrupting context, all as above described. Line 47 carries the quiesce signal for clearing current context register 32. Line 48 represents resetting the associated bit in active context register 31. In other words, the software executing in chip processor 11 merely initiates the quiescing of a context.

The general machine operations of FH circuit

37 are shown in Fig. 4. Machine step 60 represents the asynchronous receipt of interrupt requests by register 30. In a first machine cycle 61, in time coincidence with software machine step 50, circuits 18 perform the next described machine steps. At step 62 circuits 18 receive the outputs of logic-OR circuits 36 as the input signals from interrupt register 30 and active context register 31. At step 63, circuits 18 find which of the contexts, if any, represented by the received input signals, has the highest assigned priority. Then at decision step 64 (corresponding to compare circuit 39 operation), whether the highest assigned priority measured in step 63 is greater than the priority level of the current executing context is measured. If the current priority level is higher than or equal to the found priority level, then no action is taken as indicated by line 65, i.e. no valid interrupt signal is generated. When the current context's priority is not higher than or equal to the found priority, then at machine step 66 the bit position in the second rank of register 30 is reset and a valid interrupt signal is supplied over line 41 to "set up new context". Then in the next ensuing machine cycle 67, chip processor performs the known PSW swap at step 52.

Fig. 5 is a machine operations chart illustrating, in simplified form, quiescing a context. At machine step 70, chip processor determines to end a context. This context is usually the currently executing context indicated in register 32, no limitation thereto intended. Extending from step 70 is a line representing the quiesce signal on line 47. In the next machine step 71 the line 47 quiesce signal resets current context register 32. In machine step 73 the corresponding active bit in active context register 31 is reset (see line 48 in Fig. 2) by the quiesce signal travelling over line 47. Processor 11 can reset the active bit via bus 14. In machine step 74 the processor 11 is in background (lowest priority level) context. Then in machine step 75, interrupt circuits 18 again find the highest priority active context or interrupt as above described. It is to be understood that processor 11 can set register 32 during the PSW swap step 52 rather than having the setting being accomplished using electronic circuits. Also processor 11 could set current context register 32 to the background level (lowest priority) during machine step 70 but before any action is taken for setting current context register 32 by circuits 18; it is preferred to relieve processor 11 of such activities.

The machine operations of FH circuit 37 is next described with respect to Fig. 6. Operations are initiated as indicated by line 80. Read step 81 captures the outputs of logic-OR circuits 36. In the match interrupt/context with priority step 82, POR-trap bit is examined on the first pass of the oper-

ations. If it is set, then the machine operations follow path 83 to set up a valid interrupt signal by actuating encode 38 and other circuits to operate as above described. If POR-trap is not set, then the next bit position of the FH circuit 37 holding register (not shown) is sensed. If set, the corresponding register 45 is sensed for obtaining the assigned priority level. At machine step 84, the sensed priority level is compared with the contents of a one-byte register (not shown) used to store the highest level of priority yet sensed in this cycle of operation. During the first pass, the contents of the one-byte register is the highest number representing background processing. It also could be set to the priority of the current context. Compare circuit 39 only provides for a valid interrupt signal if the found highest priority is greater than the current priority level. In any event, if the compare at step 84 finds the priority of the context being examined is lower than the priority in the one-byte register, then later-described housekeeping operations occur. When compare step 84 finds the priority of the context being examined is a higher priority than the contents of the one-byte register, then at step 86 the priority of the context being examined and the context identification is stored in a one-byte register. If at compare step 84 the compared priorities are equal, then at machine step 87 the address of the context register 45 (context number, not priority) and the context register 45 address for the context identified in the one-byte register are compared. As a tie-breaker operation, the context having the lowest register 45 address (lowest context number) is determined to have the highest priority, a fixed second level evaluation of priority to allow multiple contexts to be assigned a like priority level. When the address of the context being evaluated is higher, then later described housekeeping operations occur. Otherwise, when the address of the context being evaluated is lower (higher priority), then that address and priority level are stored in a one byte register at step 88. The housekeeping operations begin from any of the steps 84, 86, 87 or 88. In step 90, it is determined whether or not more contexts/requests received from registers 30, 31 have to be examined. If not, machine operations proceed to step 64 over line 91. The next bit, identified at step 92, from the input signals is then obtained at step 81 and the loop of operations repeat until it is determined that all of the set bits received by FH circuits 37 have been examined. The found highest information is then transferred by FH circuit 37 to encode 38.

Claims

1. Data processor interrupt logic which sepa-

ately maintains stored a record of the priorities of processor contexts, which contexts are active, which context is the current executing context and which contexts are requesting to interrupt the current context and continuously compares the priorities of the current context and the highest priority requesting active context as determined by combining the stored records, validating - enabling - the interrupt if its priority is the higher.

2. A data processor including interrupt logic as claimed in claim 1 and comprising, in combination: interrupt storage storing indications of interruption signals;

active context storage storing indications of all contexts that are currently active for the processor;

priority storage storing context priorities;

current context storage storing an identification of the current context in which the processor is executing;

find logic operatively connected to the interrupt, active context and priority storages receiving the stored indications therefrom and determining which active context indicating an interrupt request has the highest priority;

interrupt validating logic connected to the find logic, to the priority storage and to the current context storage for comparing the priorities between the highest priority context indicated by the find logic and the priority of the current context for indicating a valid interrupt when the find logic indicated priority is higher than the current context priority;

logic connected to the interrupt validating logic for responding to the valid interrupt indication for interrupting the programmed processor; and

activating logic connected to all of the above recited logic for actuating such logic to determine whether a stored indication of an interruption signal represents a valid interrupt.

3. A processor as claimed in claim 2, wherein the priority storage has an addressable context register for each of the contexts, each of the addresses of the context register being a secondary interruption priority, each of the context registers storing a primary interruption priority of the context and there is provided resolution logic in the find logic responsive to two of the contexts having an identical stored primary interruption priority for comparing the addresses of the two context registers for the two contexts and selecting a context register having a predetermined address relationship of the two context registers for indicating the context corresponding to the selected context register as the context with a higher priority between the two contexts having an identical primary interruption priority.

4. A processor as claimed in claim 3, wherein the active context storage, the current context storage and the context registers of the priority storage

are addressable by the processor such that the contents of the active context, current context and priority storages are settable by programming executing in the processor.

5. A processor as claimed in claim 4, further including end context logic connected to the active context storage and to the find logic for actuating same to erase an indication of the current context from the active context storage and to actuate the find logic to identify the highest priority context or interruption signal and connected to the interrupt validating logic to make the find logic identified highest priority context, the new current context.

6. In a data processor, a method of processing interrupt signals for determining whether or not an interrupt is to be taken, the method including the machine-executed steps of:

establishing a plurality of operational contexts for the programmed processor and primary and secondary priorities of interruptions for each of the contexts;

storing one or more interruption signals which indicate requests for interruption of the processor operation for changing instruction execution to a context corresponding to the stored interruption signals, respectively;

maintaining an indication of all active contexts in which the programmed processor may or may not be currently executing instructions and identification of the current context in which the programmed processor is executing instructions; during predetermined cycles of operation of the programmed processor comparing all of the active contexts and stored interruption signals with each other including ascertaining the primary and secondary priorities of each for indicating a highest priority active context or interruption signal; and comparing the indicated highest priority active context or interruption signal with the current context in both primary and secondary priorities and interrupting the programmed processor when the indicated highest priority active context or interruption signal is higher than the priority of the current context.

7. A method as claimed in claim 6, further including quiescing the current context and selecting the indicated highest active context or interruption signal indicated context as the current context.

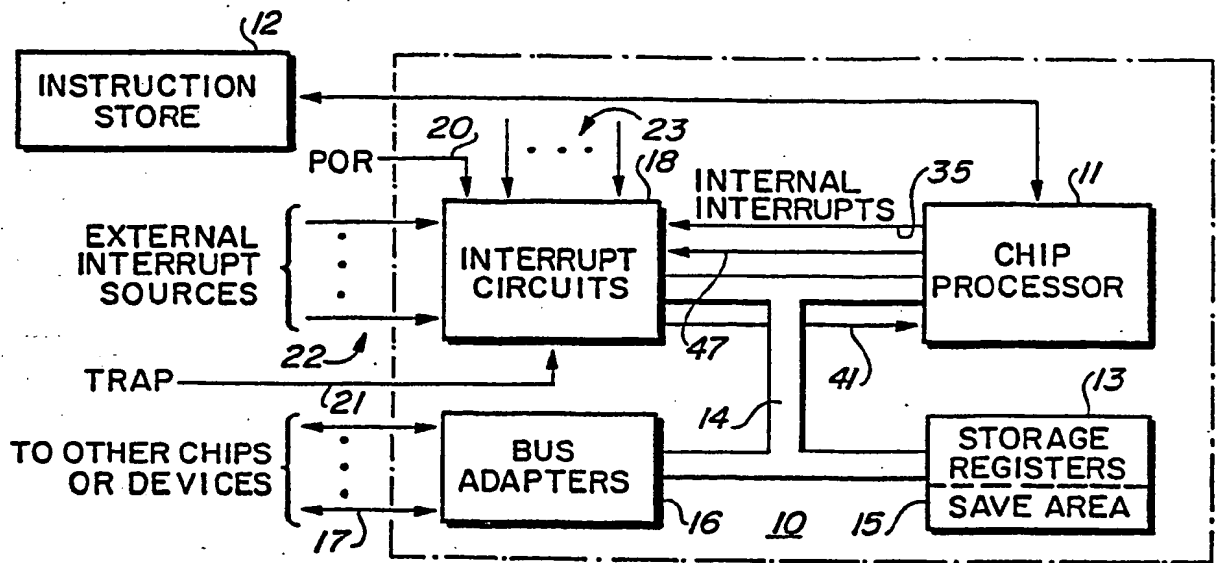


FIG. 1

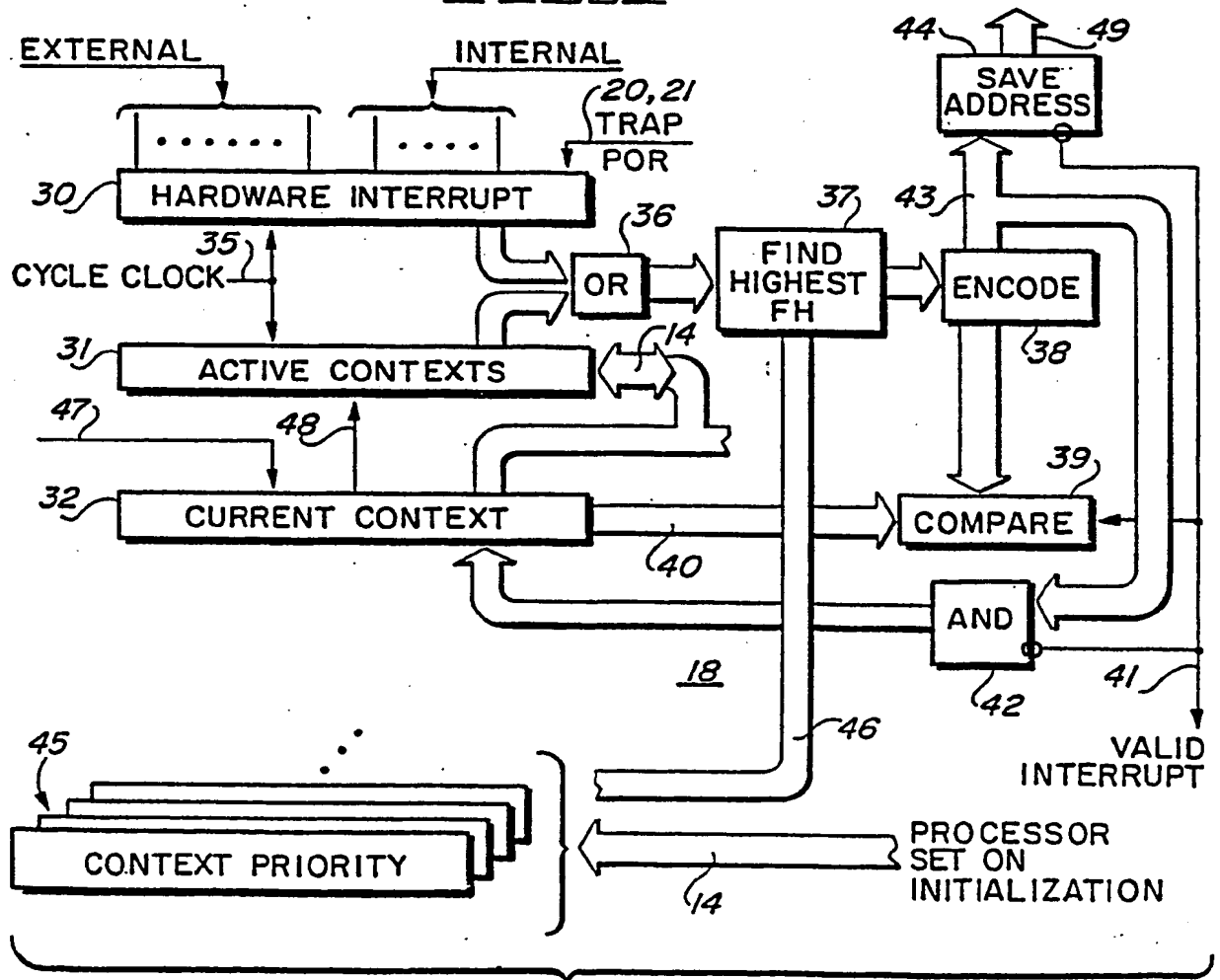
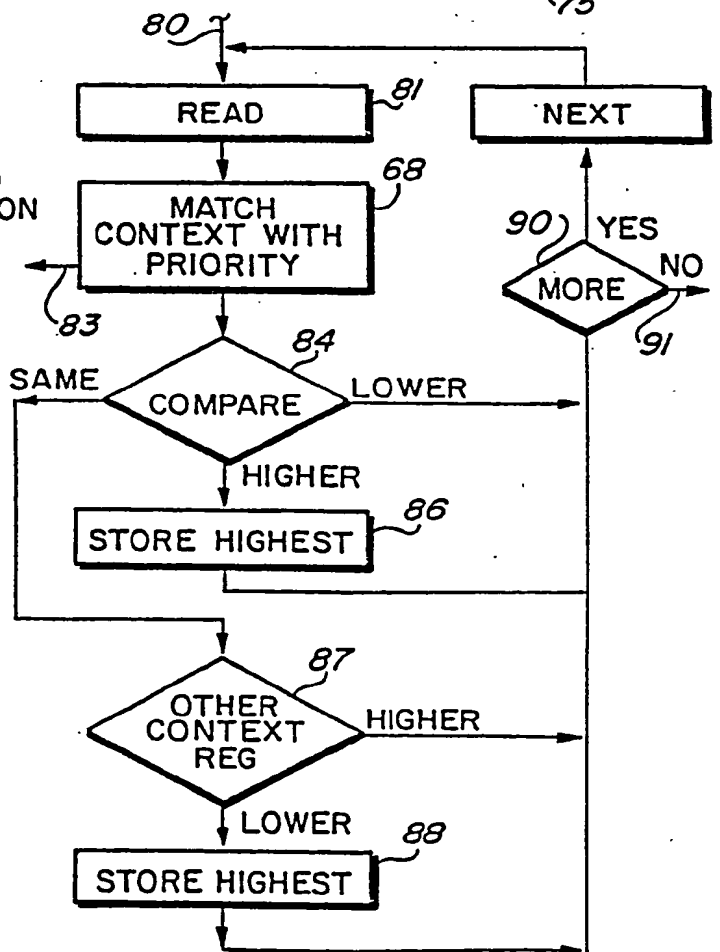
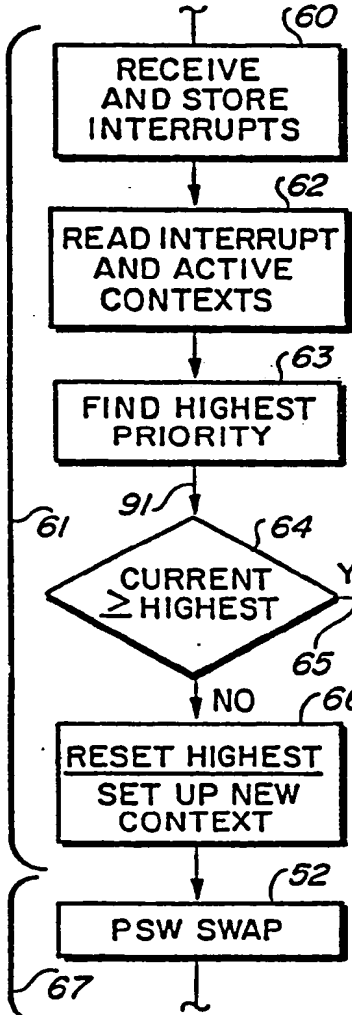
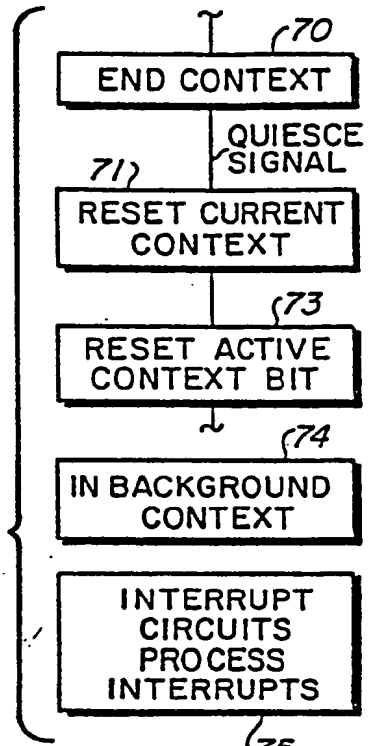
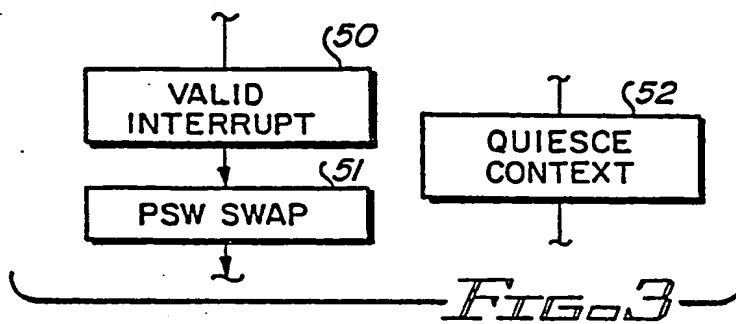


FIG. 2



This Page Blank (uspto)